Attorney's Docket No. FR920010023US1/9407-24 PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re: M. Barturen et al.
Serial No.: 09/943,563
Filed: August 30, 2001
Examiner: Chrystine Pham
Group Art Unit: 2192
Confirmation No. 2529
For: INTEGRATED SYSTEM AND METHOD FOR THE MANAGEMENT OF A COMPLETE END-TO-END SOFTWARE DELIVERY PROCESS

Date: July 10, 2007

APPELLANTS' CORRECTED REPLY BRIEF UNDER 37 C.F.R. § 41.41

This *Corrected Reply Brief* is filed pursuant to 37 C.F.R. § 41.41 to respond to the arguments raised in the *Corrected Examiner's Answer* mailed July 2, 2007. It is not believed that an extension of time and/or additional fee(s) are due. If any additional fee or extension of time is required, Appellants request that this be considered a petition therefor. The Commissioner is authorized to charge any additional fee which may be required, or credit any refund, to our Deposit Account No. 09-0457.

Reply to Arguments

The following sections provide Appellants reply to each of the arguments set forth in the *Corrected Examiner's Answer*.

I. The Element Inventory of Goiffon Does Not Store Software Code Modules

Almost every one of the arguments raised in the *Corrected Examiner's Answer* is based on an **assumption** that **actual software code modules** are stored in the Element Inventory **102** of Goiffon as opposed to just meta-data that describes such software code modules. However, as shown below, Goiffon includes numerous statements that demonstrate that **this assumption is incorrect**. Once it is realized that the system of Goiffon does not store software code modules in the Element Inventory **102**, it becomes clear that the *Corrected Examiner's Answer* fails to rebut any of the numerous grounds set forth in Appellants' *Corrected Appeal Brief* for reversal of the pending rejections.

Turning to the evidence, we start at the beginning of the Detailed Description of Goiffon where the Element Inventory **102** is introduced and described. Here, Goiffon expressly states that the **elements** that are stored in the Element Inventory **102** are **objects that store meta data** (i.e., "data about data"):

> **Element Inventory 102 . . . stores the various objects**, or "elements", that are **used to manage the code and data components** (not shown in FIG. 1) that support an enterprise. Each of the **objects stores meta-data, or "data about data"**.

(Goiffon at Col. 6, lines 58-62) (emphasis added). The Field of Invention section of Goiffon similarly states:

> This invention relates generally to an improved object management system for **tracking, cataloging, and managing data and code modules**; and, more specifically, to an object management system and method for **using objects storing meta-data to model a reusable group of code of data modules**.

(Goiffon at Col. 1, lines 26-31; *see also* Col. 2, lines 60-64, stating that the system "**catalogs** each of the code and data modules in the system for re-use using a respective **object that stores meta-data describing the respective module**"). Goiffon repeatedly explains that these meta data "objects" or "elements" that are stored in the Element Inventory **102** describe the location, type and various other attributes of the software data and code modules:

> This meta-data [in the objects stored in Element Inventory **102**] describes, among other things, the location of, and the type of, data or code that is stored within the respective component or module residing elsewhere within the system. This meta-data stored in an element also describes the various relationships that the respective data or code module has with other data and/or code modules. In this manner, the elements stored in the Element Inventory **102** serves as an index that points to, and describes, the various data and code resources . . . .

(Goiffon at Col. 6, line 63 through Col. 7, line 6; *see also* Goiffon at Col. 4, lines 1-6). A later part of this section of Goiffon then again confirms that the elements that are stored in Element Inventory **102** are meta-data that describes attributes of other code and data elements:

> Also included . . . are functions that allow the user to manage, view, and report on the elements and element relationships existing within the Element Inventory **102**.
>
> *   *   *
>
> [E]ach of **these elements includes meta-data that describes the location and function of the associated code or data element**. This meta-data will further describe the relationships that an element has with other elements . . . .

(Goiffon at Col. 10, lines 1-4 and16-19) (emphasis added). Similar descriptions regarding how the elements that are stored in the Element Inventory **102** are used to **model** the code and data modules are provided throughout Goiffon. (*See, e.g.*, Goiffon at Col. 2, lines 57-61, Col. 4, lines 3-8; Col. 11, lines 5-11).

Goiffon also **expressly, and repeatedly, states that the code and data modules are not stored in the Element Inventory 102** as assumed in the *Corrected Examiner's Answer*, but are **instead stored elsewhere in the system**. For example, in describing what the Element inventory **102** is used for, Goiffon states:

> Element Inventory **102**, which stores the various objects, or "elements", that are used to manage the code and data components (not shown in FIG. 1) that support an enterprise. Each of the objects stores meta-data, or "data about data". This meta-data describes, among other things, the location of, and the type **of, data or code that is stored within the respective component or module residing elsewhere within the system**.

(Goiffon at Col. 6, lines 59-66). Thereafter Goiffon repeatedly confirms that the elements residing in the element inventory model the code and data modules, and that the code and data modules are **stored elsewhere in the system**:

> The Properties View enables the user to view the list of attributes associated with a selected element or element type, and [sic] which are stored within the Element Inventory **102**. . . . The relationships which are provided in these views represent relationships **between the code and data modules are stored elsewhere**, and that are modeled by the elements.
>
> * * *
>
> The **Element Inventory**, which is discussed above in reference to FIG. 1, is the collection of **elements, each of which is an object storing meta-data about other code, data, or system entities residing elsewhere**. This meta-data describes, either directly or indirectly, where the respective entity resides (for example, which directory and server stores the entity).
>
> * * *
>
> [T]his **element represents an actual table of data which exists elsewhere on the system**.
>
> * * *
>
> For any of the Asset **Elements, information is also stored that describes the location of the associated code, data or system component**.

(Goiffon at Col. 11, lines 14-17 and 35-39; Col. 12, lines 58-64; Col. 19, lines 56-57; Col. 20, lines 44-46; Col. 20, lines) (emphasis added).

Goiffon also expressly (and repeatedly) identifies **where** the code and data modules are stored, namely in external host systems such as, for example, Host A **228**:

> [A]nother development **server storing code, data, or system modules represented and managed by the elements stored in Element Inventory 102**. . . . **FIG. 2** shows a network interconnection represented by Line **224** [sic] connecting a Host A **228** to Data processing System **219**. **Host A 228 includes memory 229 which stores code and data modules** (not shown).
>
> *   *   *
>
> Each element type may represent a particular type of software construct or data structure **for a code or data module existing within one of the host systems** interconnected to Object Management System **100**.
>
> *   *   *
>
> [A]n element of type "table" may be created within Element Inventory **102** to **represent a data module existing on one of the host systems** interconnected to the Object Management System **100**. . . . To restate, each element type **302** represents a type of **a potentially reusable code or data module located on various host systems** managed by Object Management System **100**.

(Goiffon at Col. 13, lines 1-6; Col. 17, lines 22-25; Col. 17, lines 41-50; *see also* Goiffon at Col. 18, lines 40-45; Col. 20, lines 51-57) (emphasis added). These teachings directly contradict the position taken in the rejections that the code and data modules are stored in Element Inventory **102**.

Appellants also note that Appellants' *Corrected Appeal Brief* included extensive quotes from Goiffon explaining that the Element Inventory **102** only stored meta-data and that the actual software was stored elsewhere, namely in remote host systems. Notably, the *Corrected Examiner's Answer* does not address any of the portions of Goiffon cited by Appellants. Appellants respectfully submit that the *Corrected Examiner's Answer* has not addressed the sections of Goiffon cited by Appellants because those sections so indisputably show that the assumption underlying the arguments presented in the *Corrected Examiner's Answer* are incorrect. Accordingly, Appellants respectfully request reversal of all of the pending rejections for the reasons set forth above, along with the additional reasons set forth in Appellants' *Corrected Appeal Brief*.

The only "evidence" cited to in the *Corrected Examiner's Answer* as supporting the assumption that actual software code modules are stored in the Element Inventory **102** are (1) step **1828** of FIG. 18B, (2) Col. 5, lines 30-42 and (3) Col. 8, lines 48-67 of Goiffon. (Examiner's Answer at 15-17). However, as discussed below, none of these portions of Goiffon support the assumption in the *Corrected Examiner's Answer* that actual software code modules are stored in the Element Inventory **102**.

A. Step 1828 of FIG. 18B of Goiffon

Step **1828** of FIG. 18B of Goiffon discusses writing an "element package" to the Element Inventory **102**. Goiffon, however, expressly states that the "Element Packages" discussed therein comprise a group of elements that model the code and data modules:

> Many reasons may exist for **grouping elements together into an Element Package**. Generally, **an Element Package will comprise elements that represent, and model, all code and data modules** that are needed to perform one or more predetermined functions.

(Goiffon at Col. 22, lines 30-34) (emphasis added). This definition makes clear that an element package is merely a group of the meta-data elements that are stored in the Element Inventory **102** that are used to model the code and data modules that Goiffon repeatedly states are stored elsewhere. (*See, e.g.,* Goiffon at Col. 6, lines 59-66; Col. 11, lines 35-39; Col. 13, lines 3-6). Thus, while step **1828** states that the "element package" is written to the Element Inventory **102**, this merely indicates that meta-data elements are stored to the Element Inventory **102**. Accordingly, step **1828** of FIG. 18B does not support the pending rejections, but instead refutes such rejections.

B. Col. 5, Lines 30-42 of Goiffon

The *Corrected Examiner's Answer* next cites to Col. 5, lines 30-42 of Goiffon as indicating that the software code and data modules are stored in the Element Inventory **102** of Goiffon. This section states:

> According to yet another aspect of the invention, existing package objects may be used during the creation of still other package objects. That is, package objects may be created that model software packages that include other software packages. According to the preferred embodiment, this is represented by forming relationships between a respective package object and other package objects. This allows interdependencies between packages of software constructs to be recorded and managed. Any number of levels of hierarchy may be created within the package object definitions.

However, Goiffon expressly teaches that the "package objects" referred to in the above quotation from Goiffon are **meta-data objects** that model a package of code and/or data modules:

> After the complete package of software constructs [i.e., code/data modules] has been identified for a given package . . ., the inclusion of each software construct in the package is recorded. . . . [T]his is accomplished by creating a package object to model the package. **The package object is similar to any of the objects for modeling a respective software construct, and stores data indicative of the software constructs included in the package**.

(Goiffon at Col. 4, lines 59-67) (emphasis added). Thus, the second section of Goiffon that is relied upon in the *Corrected Examiner's Answer* merely states that existing package objects (i.e., meta-data about a package of code and data modules that are stored elsewhere) can be used during the creation of other package objects (i.e., additional meta data about a package of code and data modules that are stored elsewhere). Appellants respectfully submit that this does not disclose storing software code modules within the Element Inventory **102**, but instead only serves to confirm that the Element Inventory **102** of Goiffon is used exclusively to store meta-data regarding software components that are **stored elsewhere**.

C. Col. 8, Lines 48-67 of Goiffon

The *Corrected Examiner' Answer* also relies on Col. 8, lines 50-53 of Goiffon, which states:

> Element Packager **118** is utilized to build the identified elements into a package that includes all of the code and data necessary to transform a group of components.

(Goiffon at Col. 8, lines 50-53). As explained in the description of "element packages" provided at Col. 22, line 29 through Col. 24, line 41 of Goiffon and in the description of the operation of Element Packager **118** provided at Col. 24, line 42 through Col. 29, line 26 of Goiffon, the above-quoted sentence refers to the creation of an "element package" that comprises elements that represent, and model, a package of code and data modules that are needed to perform a particular function. (*See, e.g.,* Goiffon at Col. 22, lines 30-34). These sections of Goiffon (as well as the numerous quotations from other sections of Goiffon provided above) make clear that the Element Package is a meta-data package that is used to model the associated code and data software components that are stored on external hosts.

The remainder of the portion of Col. 8 cited in the *Corrected Examiner's Answer* discusses how a "wrapper" may be provided that performs a transformation operation that allows, for example, the software code and data modules to operate in a different environment. The *Corrected Examiner's Answer* attempts to argue that this "wrapper" is part of the "Element Package" to suggest that software modules are stored in the Element Inventory **102**. Goiffon, however, expressly refutes this argument.

In particular, Goiffon repeatedly explains that the "wrappers" are not part of the Element Packages, but instead are something – like the software code and data modules – that may be "associated with" an Element Package:

> The above example illustrates the **manner in which Element Packages are used to develop code wrappers**. Once a basic group of code and data modules are identified using the **associated elements** as shown in FIGS. 13 and 14, the identification process does not need to be repeated. The functionality modeled by the Element Package may be used multiple times to develop multiple code wrappers. The relationships and elements included in the Element Package are hidden, and only the relationships that are of interest to the "external world" need be considered. This makes the relevant code and data relationships associated with the wrapping function easier to conceptualize. Moreover, once created, the Element Packages are available for reuse. For example, the user may include the same element "Transaction1" in more than one package, with each of the **packages being associated with a different wrapper** for a different data processing environment. That is, once the user has determined a core group of code and data modules using the element definitions, this **Element Package can be associated with different code wrappers** without having to re-verify that all the necessary code and data modules, and all of the associated interfaces have been considered.

(Goiffon at Col. 34, lines 15-37) (emphasis added). As further explained in Goiffon, the "wrapper" is a software package that is created by a software engineer in order to make the software code and data modules callable from within a different data processing environment:

> The Element Package definition of element Transaction1 can be used by a programmer to determine how a code wrapper is to be created. That is, a software engineer entirely

> unfamiliar with the interrelationships existing between the code and data modules modeled by the package need only understand the interface definition for the Element Package to design the code module which will "wrap" the transaction, thereby making the transaction callable from within a different data processing environment. Thus, the Element Package models the functions performed by the included elements in a way that simplifies the task of an engineer designing interfacing code and/or data modules.

(Goiffon at Col. 33, lines 43-54). The programmer uses the Element Package to figure out how to make this wrapper, and does not need to look to the interrelationships existing between the individual code and data modules. (*Id.*) Once the wrapper is created, it is treated like any other of the software components that are stored elsewhere, and an element is thus created for the wrapper and included in Element Inventory **102**:

> [T]he interface definition may be used to design the code module that will perform the wrapping function. Thereafter, an Asset Element may be created to describe the newly-created code module "wrapper".

(Goiffon at Col. 33, lines 55-59). Thus, Goiffon clearly demonstrates teaches that the "wrappers" are not stored in the Element Inventory **102**, but instead are merely later created software, and that an element is created and stored in Element Inventory **102** that models this software just like elements are provided that model all of the other software code modules.

For all of the above reasons, Appellants respectfully submit that Goiffon does not anticipate any of the pending claims.

II. Response to Argument Regarding Claim 7

Appellants submit that the above section responds to each of the arguments set forth in the *Corrected Examiner's Answer*. Appellants, however, provide the following further comments regarding the argument presented in Section C of the *Corrected Examiner's Answer*.

Section C of the *Corrected Examiner's Answer* argues that lines **227** and **240** of Goiffon disclose a "a sixth sub-system for recording information provided by at least one of the first through fifth sub-systems of the integrated data processing system during delivery of the software product" as recited in Claim 7. However, this argument confuses **reading and writing the elements** between Client Server **216** and Data Processing System **219** with **recording information** provided by various subsystems. In fact, the *Corrected Examiner's Answer* takes the clearly inconsistent position that importing and exporting elements comprises both the distribution of the software package by the third sub-system and the recoding of information by the sixth sub-system. In any event, neither the pending rejections nor the *Corrected Examiner's Answer* even attempts to explain what information is recorded or what sub-system the information that is recorded is allegedly provided from, and thus a *prima facie* rejection of Claim 7 has not been established. Appellants respectfully submit that this position in the *Corrected Examiner's Answer* is clearly inconsistent and cannot support a rejection of Claim 7.

III. Conclusion

In light of the above discussion, Appellants submit that each of the pending claims is patentable over the cited art and, therefore, request reversal of the rejections of Claims 1-17.

Respectfully submitted,

[illegible]

D. Randal Ayers
Registration No. 40,493

USPTO Customer No. 20792
Myers Bigel Sibley & Sajovec, P.A.
Post Office Box 37428
Raleigh, North Carolina 27627
Telephone: (919) 854-1400
Facsimile: (919) 854-1401

CERTIFICATION OF ELECTRONIC TRANSMISSION UNDER 37 CFR § 1.8

I hereby certify that this correspondence is being transmitted electronically via EFS to the U.S. Patent and Trademark Office on July 10, 2007.

Michele P. McMahan

Michele P. McMahan
Date of Signature: July 10, 2007